Measuring Mobile Broadband Networks in Europe

Task 1 (Design/Exec) report

Name of your project:	CamCoW
Your organization(s) name(s):	Queen Mary University of London
Name of contact person for report:	Alvaro Garcia-Recuero Gareth Tyson
Contact person telephone number:	
Contact person email:	alvaro.garcia-recuero@qmul.ac.uk gtyson@qmul.ac.uk

Part A. Project Summary

Please provide a summary of the main achievements and the work carried out within your project (1 page). The information provided in the summary is intended for use in public documents and reports by the MONROE project.

We have completed the development of the first deliverable in the form of a toolkit, the architecture of the solution proposed is depicted in the **Figure 1** below. Roughly, the **workload generator** code produces the experiments to run in the container in an automated fashion from a JSON file, locally or remotely from a URL. Secondly, the parser takes those outputs as input parameters and process them accordingly to launch the measurement internally into the container. Besides, the design of the solution proposed allows for further extension by adding more tools to the basic toolkit (e.g., traceIXroute, tracebox, phantomjs, etc.). In fact, we have already installed some of those tools, for reference look at the install.sh bash script in the source code but we will describe in detail the role of each of the scripts later.

The complete toolkit, except the plotting infrastructure, is embedded into the **qmmonroe** container, publicly available on Docker Hub here. Below there is an example of the type of configuration we use as **JSON config** in **Figure 2**. This shows how to pass the json parameters nested to the workload generator.

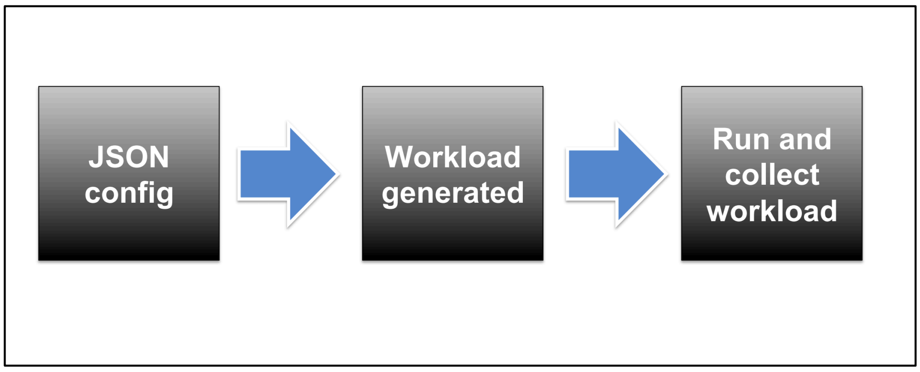


Figure 1. Architecture design of workload generator

The most relevant configurations in the JSON config we use for experiment deployment are the **name**, **probes**, **duration**, **interval** and **target** list as seen in **Figure 2**. Additionally, we provide an experiment description to each json entry or experiment.

```
"experiment": [
  {"name": "ping",
  "probes": 10,
  "duration": 60,
  "interval": 10,
  "target": ["google.com", "youtube.com", "facebook.com", "baidu.com", "wikipedia.org", "reddit.com",
"yahoo.com", "google.co.in", "qq.com", "amazon.com"],
  "description": "Ping experiment"
  },
  {"name": "traceroute",
  "probes": 10,
  "duration": 30,
  "interval": 10,
  "target": ["google.com", "youtube.com", "facebook.com", "baidu.com", "wikipedia.org", "reddit.com",
"yahoo.com", "google.co.in", "qq.com", "amazon.com"],
  "description": "Traceroute experiment"
  }
 ]
```

Figure 2. Json parameters to workload

Part B. Detailed description

This section describes the details on the experiment/extension.

B1 Concept and Objectives

Describe and explain the overall concept that forms the basis for your experiment and/or extension. Describe the main ideas, models or assumptions involved. Describe the specific objectives of your experiment and/or extension. Describe how your work contributes to the MONROE objectives. (1-2 pages)

The objective of the project is to build a toolkit or framework to perform test-harness of the infrastructure offered by Monroe regarding Mobile Broadband Networks in the Wild. The deliverables promised by Queen Mary at the initial proposal are the following:

- Task 1: Designed and executed basic/core experiments using our framework in monroe-bench. However, we need to embed the *dnslib.py* into our framework to map each interface to the operator DNS resolver as discussed in the workshop in January 2018. We also performed some preliminary Analysis of the data by automatically collecting it from the Monroe platform using the monroe-cli and parsing the data using the draft scripts developed by Dr. Álvaro García-Recuero (*ping-plot.R* and *traceroute-plot.R*).
    - Prototype developed by end of 2017 by Dr. Álvaro García-Recuero.
    - Plotting infrastructure developed at beginning of 2018 by Dr. Álvaro García-Recuero.
    - First round of experiments done by February 2018 by Dr. Álvaro García-Recuero, however we have not been able to record IXP interconnection points yet. That should be fairly easy and in fact the traIXroute software is embedded into the container once enabled in the *install.sh* script. MaxMind or geo-mapping of results with online sites can be easily performed with the resulting traces of our experiments.
- Task 2: Analysis and Experiment revisiting to begin in February/March 2018: TODO

- Task 3: Feedback and reporting has been continuous over the length of the project: TODO
- Task 4: Dissemination starts in April 2018 or so.

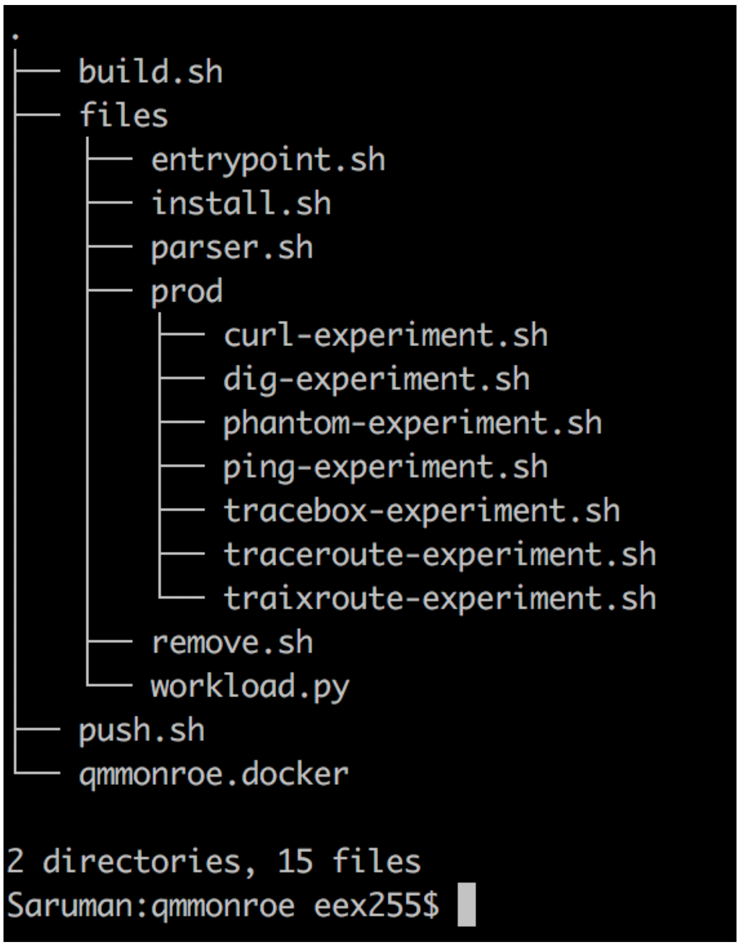
B.1.0. Our contributions

The repositories with the test-harness and the container are available at the following address, https://github.com/algarecu/monroe-bench.git

B.1.1. Container files

Firstly, the **monroe-bench/src/qmmonroe** provides the container code.

The container files are located in **src/qmmonroe,** which provides all the relevant container configuration and a .Dockerfile for deployment to the platform. Please see **Figure 3** which summarizes the code structure of the container files. Next, we explain the role of each of the files in this folder and how they interact with the container.



```
.
├── build.sh
├── files
│   ├── entrypoint.sh
│   ├── install.sh
│   ├── parser.sh
│   ├── prod
│   │   ├── curl-experiment.sh
│   │   ├── dig-experiment.sh
│   │   ├── phantom-experiment.sh
│   │   ├── ping-experiment.sh
│   │   ├── tracebox-experiment.sh
│   │   ├── traceroute-experiment.sh
│   │   └── traixroute-experiment.sh
│   ├── remove.sh
│   └── workload.py
├── push.sh
└── qmmonroe.docker

2 directories, 15 files
Saruman:qmmonroe eex255$
```

Figure 3. Code Structure of container

List of files in **Figure 3** and their description:

- Root folder: this is the folder with the actual container files deployed to the Monroe platform, build.sh, push.sh and qmmonroe.docker.

- Subfolder ***files***:
  - This is the folder with the actual container files deployed to the Monroe platform.
    - In *workload.py* we generate the experiment workload to dispatch to the container.

- In *parser.sh* we iterate through the lines generated by the *workload.py* to run each experiment sequentially and avoid interference among several runs of each of them.
- In the script *entrypoint.sh* we take each of the metadata that needs to be considered for the running of the experiment correctly. Here we also look at how many interfaces we have in the node where the container is deployed by extracting it from the shell.
- The scripts *install.sh* and *remove.sh* install and remove the appropriate containers. Note each can be called individually, adding only 1 layer to the Docker container.

❖ Subfolder ***files/prod***:
- ➢ Contains the experiment files present for production deployments. These are the list of scripts to run experiments used in the *entrypoint.sh*, which can be itself extended following the same logic to accommodate new tools/experiments accordingly with minimum effort.

The *.Dockerfile* provides the code that sets up all the tools we list. It first configures the permissions on the files that need to execute.

It first copies the scripts in *files/prod* that will run the experiments into a specific location into the container. Then it executes *install.sh* to add the packages required for the typo of experiment. Once that is done, calls the script to execute the actual experiment using the line:

```
CMD ["/bin/bash", "-c", "/opt/monroe/parser.sh"]
```

The above line by triggers the **entrypoint.sh** script with the parameters generated by our *workload.py* automatically. This will iterate through as many commands were generated in the workload.

B.1.2. Cli files

❖ Subfolder ***auth*** (not shown)**:**
- ➢ It should contain the .crt files extracted from the .p12 file for password less login into the platform. A new certificate is required for this since the existing is expired. **Do not commit the certificate in this repo!**

The source files here are simply a wrapper to the cited monroe-cli project as to deploy experiments from the command line automatically and by passing our script configuration also here. We would need to integrate this into the monroe-cli tool if there is interest. I am in contact with Ana Custura from Aberdeen University for that.

The code lives in **monroe-bench/src/cli** and **monroe-bench/src/deploy_experiment.py** are the wrapper we have developed around the cite monroe-cli tool provided by Ana Custura (Aberdeen University). This can be extended accordingly, and it needs to be further tested using real access to the platform in order to verify the completion of a comprehensive test-harness that automates the measurement of Internet protocols we are interested in over the Monroe platform.

B2 Experimental Set-up and Results

Describe the details of your experiments and/or HW/SW extensions. How was your experiment set up? What are the main results and conclusions obtained? How was any SW/HW extensions designed? How can your extensions be used by future experimenters? (4-8 pages)

We started with the first round of experiments in December 2017 approximately. We have managed to automatically deploy and collect experiment generated raw data from the Monroe platform. In the first experiments we successfully deployed we obtained data for *ping* and *traceroute*. Here we show an instance of these experiments and make some preliminary observations as promised in the deliverables of the project (see Gantt chart in Figure 1 of **CaMCoW-updated.pdf**).

- Ping experiments

Preliminary results below show a somehow surprising difference in response time for a same destination when multiple interfaces are used in the deployed Monroe node. In **Figure 3** we observe that the wired interface (eth0) exhibits great performance. In fact, we verified that in that particular experiment, there was a significant difference between wired and mobile interfaces. This should be an interesting avenue to research in more detail.

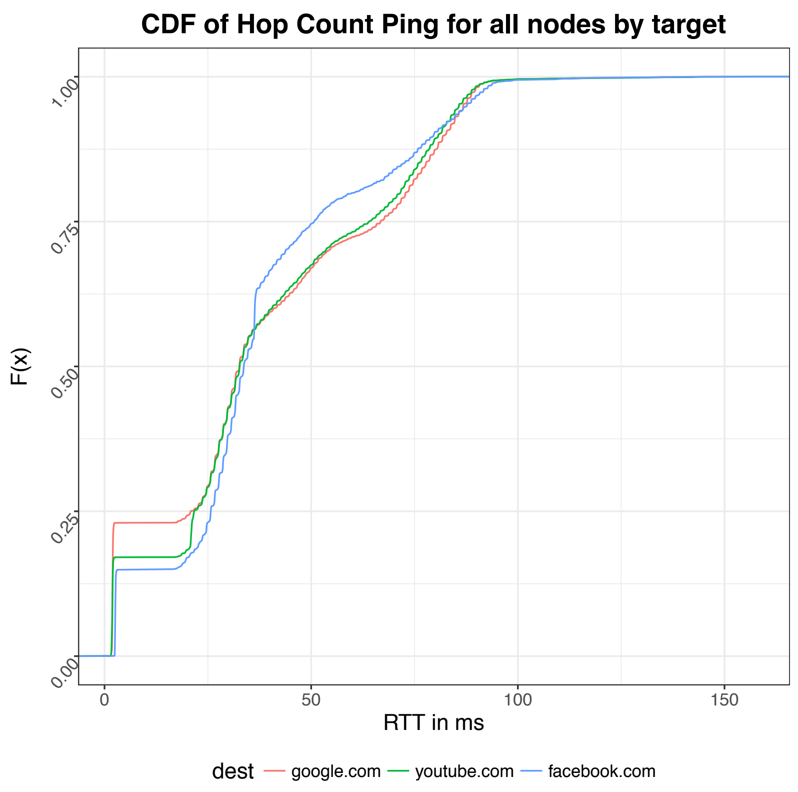


Figure 4. All targets in all nodes selected

Next, we see that when grouping by destination that roughly, the results are very similar. The results in **Figure 4** show that performance is very consistent across websites for each of nodes as we plot here the individual lines representing each deployed container for the experiment.

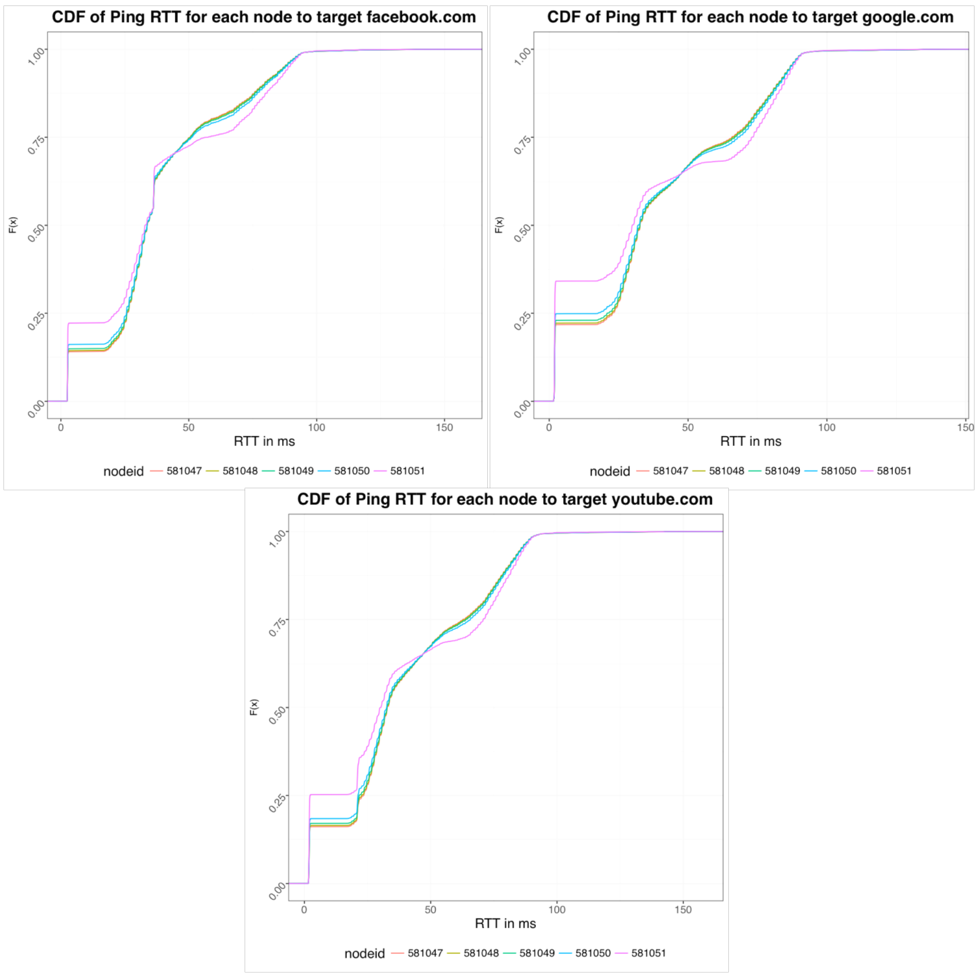


Figure 5. Subset of Alexa top-10 targets for each node

➢ Traceroute experiments:

To be updated…

➢ Curl experiments

To be updated…

- <u>Dig experiments</u>

To be updated…

B3 Lessons learned

Describe the key lessons learned from your project that can help guide future experimenters or the further development of the MONROE platform. (1-2 pages)

- **Problem 1**: we did try to run a large-scale experiment with ping for 12 hours on 48 nodes with the result being unsuccessful. We observed that the platform cannot deploy the container in all nodes, which indicates an issue but not related to our container image in this case. It is also impossible to automatically allocate specific quotas of storage and data allowances to each container. This means it is necessary to manually calculate estimates on a per-instance basis. This is particularly challenging if the duration of the experiment is changing per-run (if, for example, an experiment generates 1MB per minute, a 12-hour experiment will require much more storage than a 1-hour experiment).
  Recommendation 1: It would be beneficial if correct storage and data quotas could be computed for each experiment, based on an estimate of how much its output takes. This could be done when a container or set of containers are being deployed by assessing the implications of the scripts. This would avoid quotas being exceeded and warnings before the experiment deployment.

- **Problem 2**: launching experiments in the platform relies too much on a platform-centric configuration. We are more interested in a container-centric configuration for this. As far as we know, there is a JSON configuration file that can be read by the container/platform, but no custom field dedicated the usage of custom variables.
  Recommendation 2: there should be a dedicated environment variable/s to pass arguments from the API to the Monroe container. However, we can pass a JSON URL already with no issue.

- **Problem 3:** The account creation procedure at https://authority.ilabt.iminds.be is needlessly complicated. On creation, accounts need to be verified with a code sent via e-mail. However, once the web browser is closed, there is no way to come back to the verification page. Users have to actively wait for the email to arrive to then copy and paste the verification code immediately, while the browser windows with the registration is still open. Accounts are only created once the verification code is entered. As long as no verification code is entered, it is possible to register with the same username and password again. While this is not a problem per-se, it is counterintuitive to understand how the system handles account creation. In our case, once we successfully managed to enter the verification code (in the roughly fifth attempt), we got errors regarding that our password was wrong and our account already existed. Nevertheless, we could login fine with the credentials afterwards. This indicates that the system works but not without intricacies.
  Recommendation 3: It would be useful to make the process transparent to the end user and provide less instructions to achieve the same goal so that the resulting registration is less prone to error by the user.

- **Problem 4:** In the user interface of Monroe, the experiment submit button does not work on Firefox 52.0.2 (64-bit) on Mac OS X Sierra. There are also problems for that when using the Safari browser. However, in both it also does not show any error message about the problem. It was quite frustrating trying to figure out what the error was, when finally switching to Google Chrome solved the problem. It is not intuitive that in the Status page the individual rows can be clicked on to see more details. Also, copying the SSH tunnel connect line is tricky, as clicking in the text collapses the experiment data and effectively hides the command line again.

Recommendation 4: we suspect this is due to something to do with the use of a self-signed certificate on the browser. We suggest creating properly signed certificates using globally recognized Certification Authorities. Actually, this seems to have updated now.

- **Problem 5:** We could not run the monroe-cli tool on a Mac with OS X Sierra operating system. The tool is not compatible with OpenSSL, which is what Mac OS X compiles against by default. *gnu-tls* is tricky to compile in Mac OS X if not impossible (depends on the OS X verion too) and not supposed to be used anyway. Simply put, it seems like the OpenSSL standard policies for certificate authentication do not play well with the MD5 signed certificates used by the platform.
  Recommendation 5: Need to install Debian VM for operating the monroe-cli command line tool.

- **Problem 6:** We noticed a failure in container deployment after updating our container image for adding more software to our toolkit. It turns out that adding more software into the container, we are unable to deploy the same set of experiments than before ran correctly. This is either, due to the fact that the platform times out while the container is updating its software in the node at deployment time or because the maximum size of the container has exceeded an upper limit we are not aware of.
  Recommendation 6: It would be highly beneficial to highlight the storage quota a container image can reach not the account only, and what other limits we should be aware not to hit (e.g., time-out, etc). Also, there is no information about how to update a container image for Monroe specifically, but it is possible to export the updated container as a brand-new image to the Docker hub. We shall test if this resolves our problem, otherwise there is an upper limit for the image indeed.

- **Problem 7:** The scheduler does not wait for synchronization between the nodes. Each node fetches and executes its own containers. Indeed, if you desired that type of synchronized execution, you would have to provide for it yourself. This means we may end up with the head and tail of some log files rendered unusable for measurements.
  Recommendation 7: There should be a way of synchronising containers before the experiment kicks-off for correctness of the measurements, etc. For example, using a master-slave synchronisation solution for the above will work. Alternatively, users should be instructed to deploy their own synchronisation mechanisms in a comprehensive way and possibly through the platform rather than using a homemade solution.

- **Problem 8:** Storage limits.
- **Recommendation 8**: we should embed most of the tools or packages we need to use into the container installation or alternatively ask the *MONROE* consortium to install them into the *monroe/base* container to simplify things and not run into issues of storage quota limit when deployment experiments.

B4 Impact

Describe the impact of your project, both for your own organization, for the MONROE consortium and for external actors. How will you use your project results and the MONROE platform in your future activities? How can your project results contribute to the sustainability and further development of the MONROE platform? What research and/or commercial opportunities have been created or can be foreseen in the future?

Please list any publications, presentations or other dissemination activities of importance. (1-2 pages for the description, excluding the list of dissemination activities).

Datasets currently located at local user account on our machine,
Public/datasets/

Results currently located at,
Public/results/

Eventually should be moved to an online location such as the following or similar for exploitation and publication,

http://www.eecs.qmul.ac.uk/~alvarogr/results/
http://www.eecs.qmul.ac.uk/~alvarogr/datasets/

Part C. Feedback to MONROE

This section contains valuable information for the MONROE consortium and describes the associated partner's experiences while performing the experiment or while implementing the extension starting from the available software and hardware. The MONROE platform has evolved during the course of the project. ***Please describe your experiences in relation to the present platform****.*

To have a fully working test-harness, attention should be paid to the cited monroe-cli tool. That is the only way to programmatically and automatically deploy/collect experiments and avoiding human intervention in the Monroe web interface. If possible, a set of configurations should be hosted in a subfolder of the monroe-bench project and then use them at each one's own discretion to launch autonomous experiments and even chain one after each other.

Finally, we are interested in integrating some of the specific parameters used for the experiments we run with Monroe. Depending on whether we want to run experiments with one interface or several ones, location, etc. Some of those options are already provided in the monroe-cli tool. However, the tool does not provide a list of commands or protocols than can run from the container/s into the Monroe platform. The latter is exactly what we will provide in our toolkit.

C.1 Resources & tools used

Please list the MONROE resources and tools you have used for your project and describe how they were used.

https://github.com/ana-cc/monroe-cli

https://github.com/algarecu/monroe-bench

C.2 Feedback on experimenting in / implementing extensions within MONROE

Feedback on a number of features of MONROE is requested below. When rating the usefulness, easiness etc., please use 5 as the highest score for all questions (indicating that a feature was very useful, very easy to use, etc.). ***Please base your ratings on how the MONROE platform is presently working.*** *If you have no experience with a particular feature asked about, please indicate this in the open ended part of the question.*

How would you rate your overall impression of the MONROE testbed? (Rate on scale from 1-5) : Please provide a paragraph describing your overall impression of the testbed.

Good, 4.

How useful is the User Manual? (Rate on scale from 1-5) : Please provide any corrections / suggestions for improvement of the User Manual.

Fair, 3.

Overall how easy is it to run experiments on the MONROE testbed? (Rate on scale from 1-5): Please provide any comments / suggestions for improvement of the overall process.

Fair, 3.

How easy is it to reuse experiment templates and Docker containers provided by MONROE for your experiments? (Rate on scale from 1-5): Please provide any comments / suggestions for improvement of the usability of MONROE EaaS features.

Fair, 3. We did not use this.

How useful is it to reuse experiment templates and Docker containers provided by MONROE for your experiments? (Rate on scale from 1-5): Please provide any comments / suggestions for making MONROE EaaS features closer to experimenter's needs.

Fair, 3. We did not use this.

How well does the Scheduling interface work? (Rate on scale from 1-5): Please provide any comments / suggestions for improvement of the scheduling interface.

It works well, however there a number of things that be improved as we already mention in the recommendations.

How well does the Docker build process work? (Rate on scale from 1-5): Please provide any comments / suggestions for improvement of the Docker build process.

It works but there are some limits to how big the container can be, roughly no more than 1GB due to eth0 download limits.

How useful is the MONROE meta data to you? (Rate on scale from 1-5): Please provide any comments / suggestions for improvement in the meta data provided.

We have barely used this but we planned to do launching a curl to localhost on port 8888 to retrieve GPS, modem, location, operator information. The following command fetches node metadata but it is yet to be proven to work or tested,

```
curl -s http://localhost:88/modems | jq > /monroe/results/node-metadata.json
```

How easy is it to access the MONROE meta data (Rate on scale from 1-5): Please provide any comments / suggestions for improvement in how meta data is provided.

Same as above.

How easy is it to access and use MONROE open data (Rate on scale from 1-5): Please provide any comments / suggestions for improvement in how open data is provided.

Historical logs are available in the web interface provided you have a certificate to access the platform.

How well does the MONROE hardware meet your requirements (Rate on scale from 1-5): Please provide any comments / suggestions for improving the hardware.

Fairly well. It is worth mentioning that we did not have any specific requirements for hardware but we may request to the Monroe consortium to install some nodes in African networks to measure latency with a distant node that possibly will reveal insights on IXP/ISP connectivity in these region, thus being able to measure interconnection latencies and routing strategies of the Content Delivery Network providers at countries with an Internet infrastructure in development.

C.3 Other suggestions

Please provide any other comments or suggestions you may have on how to improve the MONROE hardware or software, how to improve the interaction between MONROE and its external users or how to improve any other aspects of MONROE.

We have provided a list of recommendations already above.